John von Neumann Institute for Computing

Programmable Architectures for Realtime Music Decompression

Martin Botteck, Holger Blume, Jörg von Livonius,
Martin Neuenhahn, Tobias G. Noll

published in

Parallel Computing: Architectures, Algorithms and Applications ,
C. Bischof, M. Bücker, P. Gibbon, G.R. Joubert, T. Lippert, B. Mohr,
F. Peters (Eds.),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **38**, ISBN 978-3-9810843-4-4, pp. 777-784, 2007.
Reprinted in: Advances in Parallel Computing, Volume **15**,
ISSN 0927-5452, ISBN 978-1-58603-796-3 (IOS Press), 2008.

http://www.fz-juelich.de/nic-series/volume38

Programmable Architectures for Realtime Music Decompression

Martin Botteck[1], Holger Blume[2], Jörg von Livonius[2], Martin Neuenhahn[2], and Tobias G. Noll[2]

[1] Nokia Research Center
Nokia GmbH, Meesmannstraße 103, 44807 Bochum
E-mail: martin.botteck@nokia.com

[2] Chair for Electrical Engineering and Computer Systems
RWTH Aachen University, Schinkelstrasse 2, 52062 Aachen
E-mail: {blume, livonius, neuenhahn, tgn}@eecs.rwth-aachen.de

Field Programmable Gate Array Architectures (FPGA) are known to facilitate efficient implementations of signal processing algorithms. Their computational efficiency for arithmetic datapaths in terms of power and performance is typically ranked better than that of general purpose processors. This paper analyses the efficiency gain for MP3 decoding on an FPGA. The results of an exemplary FPGA implementation of a complete MP3 decoder featuring arithmetic datapaths as well as control overhead are compared concerning performance and power consumption to further implementation alternatives. Furthermore, the results are compared to requirements for a deployment in portable environments.

A Introduction

Music playback has become a distinguishing and indispensable feature in modern mobile communication devices. Mobile phones with several Gigabytes of built in storage capacity have been released to the market. Consequently, achievable playback times are no longer determined by the amount of available tracks or storage but by power consumption of the playback function. In a previous study[1] the power consumption of general purpose processors such as the ARM 940T processor core which is frequently used in embedded applications has been modeled. As an application example typical signal processing applications such as audio decoding algorithms like MP3 and AAC were regarded. Although such embedded processors yield attractive low power consumption the absolute values are still too high to support radical improvements for future mobile devices. Therefore, further implementation alternatives have to be studied retaining the flexibility of such platforms in order to be able to e.g. adapt to changes in coding standards etc. Generally, modern digital signal processing applications show a rapidly growing demand for flexibility, high throughput and low power dissipation. These contradicting demands however can not be addressed by a single, discrete architecture (e.g. general purpose processor). A typical approach to solve this problem combines different architecture blocks to a heterogeneous System-on-a-Chip (SoC). Figure 1 shows the design space for various architecture blocks like general purpose processors, field programmable gate arrays (FPGAs), physically optimised macros etc. in terms of power-efficiency ($mW/MOPS$) and area-efficiency ($MOPS/mm^2$) for some exemplary digital signal processing tasks[2]. While programmable processor architectures provide less power- and area-efficiency than other architecture blocks they have a

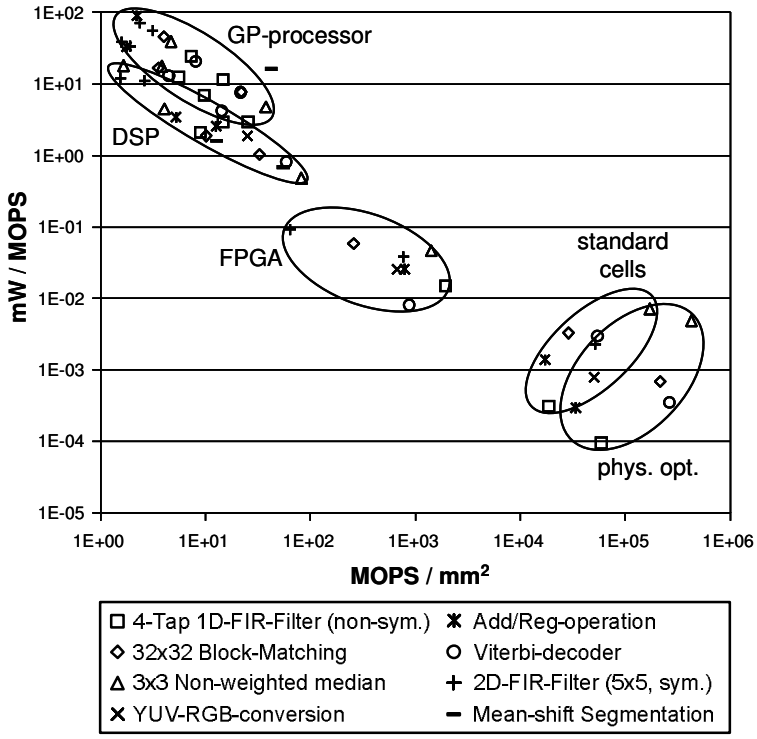

Figure 1. Design space for different architecture blocks[2]

high flexibility (not depicted in Fig. 1). The best power- and area-efficiency is achieved by physically optimised macros. However, they provide no flexibility.

FPGAs present an attractive compromise between these two extremes as they allow for highly parallelised implementations while preserving in-system reconfigurability at moderate implementation cost. The optimum partitioning of a system according to the available architecture blocks on a SoC is a difficult task that requires good knowledge about the design space. Based on these results it is important to study the resulting relations for applications that consist of number crunching components such as typical arithmetic datapaths and which also feature significant control overhead being implemented on processors as well as on FPGAs. MP3 decoding is such an example which features datapaths as well as control parts. Furthermore, it is a key application for commercial mobile devices requiring lowest power consumption and highest area efficiency. This paper is organised as follows: Section B describes the implementation of a MP3 decoder on a state-of-the-art FPGA. Section C outlines results from simulations and measurements. Section D attempts to show opportunities for further improvements by tailoring the FPGA structure for such a processing task. A conclusion is given in Section E.

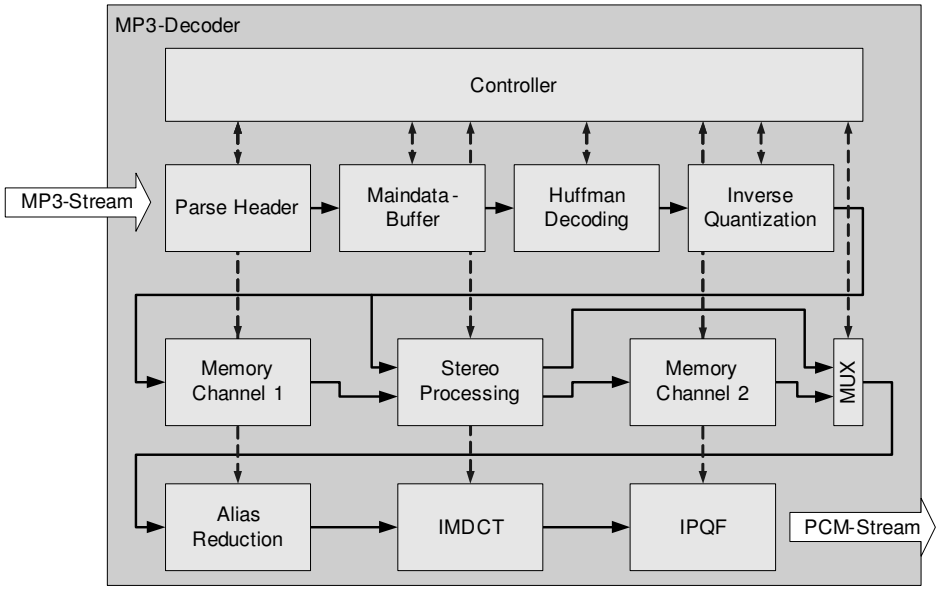

Figure 2. MP3 decoder processing blocks

B MP3 Decoder Implementation Overview

As being mentioned in Section A, the decoding of MP3 encoded music data is a key task for mobile devices. Here, the possible implementation results in terms of Audio data. The bitstream is composed of frames containing the compressed audio information[3]. The MPEG-2 standard[3] further describes a set of processing steps for decoding. These include e.g. Huffman decoding, inverse Discrete Cosine Transform, Quadrature Filters and some more.

The described implementation (see Fig. 2) realises each of these processing steps as a separate functional block. All these blocks work in parallel as part of a processing pipeline. Furthermore, separate processing of the left and right stereo channels is done in parallel as well. Especially in case the music is not encoded using the ”joint stereo” mode processing in both channels is completely independent from each other. Because of data dependencies it rendered rather difficult to identify further opportunities for parallelisation inside the processing blocks themselves. Instead, synthesis of the algorithm was optimised for maximum usage of those blocks inside the FPGA that are specialised for multiply/add/accumulate. In fact, 126 of these DSP blocks[5] work in parallel to execute the decoding algorithm. Consequently, the processing pipeline will operate at a rather low clock frequency ($<$5 MHz).

Special attention was paid to those blocks being most complicated to implement: the inverse DCT (IMDCT) and the inverse pseudo QMF (IPQF). These modules involve ROM tables for coefficient mapping (IMDCT) and a rather large ring buffer (IPQF). Implementation of stereo processing covers the basic parts only (dual mono, dual channel stereo). Joint Stereo decoding would substantially increase implementation effort and also power consumption; its absence does not adversely affect the observations described in Section D.

C Simulation and Implementation Results

The MP3 decoder has been synthesized for an Altera Stratix II FPGA (EP2S30F484C3, speed grade 3)[5] using the Altera Quartus II V5.0[6] software environment. This FPGA provides programmable logic blocks (ALUTs) and dedicated DSP blocks to realise functionality typical for signal processing algorithms. Further, there are several types of memory available which can be associated to specific processing stages.

Resource-type	Used / Achieved	Utilization
ALUT	18,701	68 %
Memory-Bits	334,592	24 %
DSP-Blocks	126	98 %
f_{max}	4.7 MHz	–

Table 1. FPGA resource usage for MP3 decoder

Table 1 shows the amount of resources from this FPGA component that were needed to perform the desired functionality. The implemented VHDL-Code has been functionally simulated with Mentor Graphics ModelSim (Version SE PLUS 6.1a). Results for processing a MP3 file with 8 seconds length are summarized in Table 2: Obviously, the IPQF block needs the highest amount of computation power; next in line followed by the IMDCT both of them consuming a magnitude more cycles than each of the other components. A similar effort balance can be seen from the resources used per component.

Functional block	Cycles used	Time $[ms]$
Parse Header	81,969	1.74
Maindata Buffer	1,798,298	38.26
Huffman Decoding	1,770,248	37.66
Inverse Quantization	2,013,002	42.83
Stereo Processing	352,512	7.50
Alias Reduction	738,090	15.70
IMDCT	16,845,606	358.42
IPQF	31,848,272	677.62

Table 2. Processing times for decoding 8 seconds of MP3

Power consumption figures have been determined using the power estimation feature of the Quartus II software[6]. Generally, there are two main effects that contribute to the total power consumption of the chip:

- base power dissipation in the clock network, circuit leakage current, I/O pads, internal regulators,
- dynamic power consumption determined by the actual amount of processing elements and memory in use.

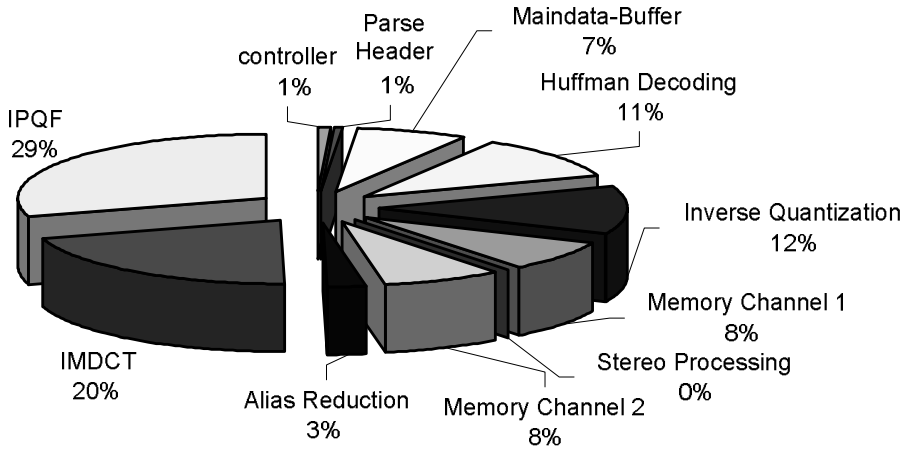

Figure 3. Dynamic power consumption per processing block

As can be seen from Table 3 total power consumption is largely determined by the base power dissipation of the chip even at a rather low clock frequency. In order to achieve these results we tried to minimize the number of registers and combinational logic. Therefore, we reduced the number of pipeline registers to a minimum, since the timing requirements can be fulfilled with a low clock frequency. Another effort to reduce the implementation costs was to optimize the tool settings of the development software, e. g. optimizing for area and using register retiming.

It is possible to further reduce the clock frequency by parallelization of the MP3 decoding on channel and frame level. This would lead to an increased controller complexity and the duplication of most of the MP3 decoder processing blocks (see Fig. 2). The positive impact on the power consumption of the clock frequency reduction interferes with the highly increased area demands. Here, the influence of the increased area on the power consumption is dominating, since the clock frequency is already low. Therefore, we did not apply parallelization of the MP3 decoding on channel and frame level.

Power consumption $[mW]$	
Dynamic P_{dyn}	27
Static P_{stat}	401
Total P_{total}	428

Table 3. Power consumption for the given implementation

Figure 3 confirms that the IPQF and IMDCT are by far the most power consuming processing blocks in the implementation.

D Evaluation of Results

Power consumption of this implementation contains a fair amount of contributions from unused components in the chip. In order to evaluate suitability for integrating such a component into a mobile device some scaling might be applied. In the deployed FPGA (Altera Stratix II FPGA (EP2S30F484C3, speed grade 3)[5] 32% of the available ALUTs remain unused. Assuming a (hypothetical) FPGA providing (nearly) the number of ALUTs needed for the design, static power consumption P_{stat} given in Table 3 will be reduced by about 32% to

$$P_{stat} \approx (1 - 0,32) \cdot 400\ mW = 272\ mW \tag{D.1}$$

As the clock-frequency has a major influence on power consumption an exact (dynamic) adjustment of the clock frequency would also have a positive effect on static power consumption. In the example simulated in Section C eight seconds of MP3-data are decoded in 7.17 seconds. Assuming a timing safety margin of about 5 % for decoding the eight seconds MP3-data will require a processing time of

$$7.17sec \cdot 1.05 \approx 7.5sec \tag{D.2}$$

While meeting these timing requirements, the clock frequency can be reduced to

$$f_{max} = \frac{7.5\ sec}{8\ sec} \cdot 4.7\ \text{MHz} \approx 4.4\ \text{MHz} \tag{D.3}$$

With this reduced clock frequency static power consumption P_{stat} dominated by the clock network will also be reduced.

$$P_{stat}^{'} \approx \frac{4.4\ \text{MHz}}{4.7\ \text{MHz}} \cdot 272\ mW \approx 254\ mW \tag{D.4}$$

The dynamic power consumption will not be affected here as clock-gating is already applied meaning that the clock is switched off after the MP3-data has been decoded. Thus, total power consumption is obtained by

$$P_{total}^{'} = P_{stat}^{'} + P_{dyn} \approx 254\ mW + 27\ mW = 281\ mW \tag{D.5}$$

Especially for the largest processing blocks (IPQF, IMDCT) further design optimisation might lead to a further reduction in clock frequency; improvement estimates at this stage remain rather speculative though. A comparison of several alternative implementations given in Table 4 shows that its power consumption makes an FPGA-based MP3 decoder no attractive alternative to embedded low power processors like the ARM 940T[1,7]. The table also lists a reference ASIC implementation from literature. It's worth mentioning that the reported power consumption value refers to only the IPQF stage being the most power consuming processing block[8].

[a] Power consumption scaled to 0.09 μm technology.
[b] Optimized power consumption including e. g. FPGA size adaptation (see Equation 5).
[c] MP3-decoding @44kHz, only subband-synthesis (IPQF).

Architecture	Chip	Technology [μm]	Power consumption [mW]	Scaled power consumption [mW][a]
RISC[1]	ARM940T	0.18	145	18
FPGA	Stratix II	0.09	428 (281[b])	428 (281[b])
ASIC[8]	—	0.35	3 [c]	—
ASIP[9]	Xtensa HiFi 2 Audio Engine	0.09	30	30
DSP[10]	VS1011e	—	40	—
DSP[11]	—	0.35	165	3
ASIP[12]	AT83SND2CMP3	—	111	—

Table 4. Comparison of MP3 decoder implementations on various architectures

E Conclusions

An FPGA implementation of a MPEG-1/2 Layer III standard decoder has been presented. The resulting power consumption however is far inferior to solutions on embedded processors optimized for low power consumption and well away from the requirements for integration into mobile devices. Computational requirements of the MP3 decoding algorithm are not high enough to fully utilise the computational capacity of this FPGA. The MP3 decoding algorithm does not provide enough inherent parallelism in order to achieve attractive implementation figures on a FPGA. For such low computational requirements serialized implementations on programmable processors are better suited. FPGA based implementations remain attractive for computational intensive applications involving highly regular datapaths with a high potential for parallelism. Apparently, the MPEG-2 standard prescribes algorithms for decoding that do not have such properties.

In order to evaluate suitability of such an implementation for a mobile device the total power consumption is the most important figure to consider. Modern mobile devices typically are equipped with 1000 mAh/3.6 V batteries. Such a battery would be completely drained by the FPGA implementation alone in less than 8 hrs. This rough calculus does not include additional contributions to the total consumption for ”listening to music” by voltage regulators, A/D conversion, power amplification and user interface operation. Further, innovating implementations shall radically increase listening times acknowledging the large music libraries which can be found on today’s music products. Typical listening times today are in the range of 10 to 30 hrs (e.g. iPod, Sansa) with storage capacities up to 30 or 60 GByte. Consequently, the total power consumption for listening to music should be well below 30 mW in order to achieve satisfactory listening times. This leaves considerably less than 10 mW for the decoding signal processor taking account of regulators and headphone amplification etc. Embedded processors are likely to achieve this target, FPGA implementations are far away from it.

References

1. H. Blume, D. Becker, M. Botteck, J. Brakensiek and T. G. Noll, *Hybrid functional and instruction level power modeling for embedded processor architectures*, in: Proc. Samos 2006 Workshop, Samos, Greece, 17–20 July, LNCS **4017**, pp. 216–226, (Springer, 2006).
2. T. G. Noll, *Application domain specific embedded FPGAs for SoC platforms*, invited survey lecture at the Irish Signals and Systems Conference 2004 (ISSC'04), (2004).
3. ISO/IEC 11172-3, Information Technology, *Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s, Part3: Audio*, (1993).
4. M. Bosi and R. E. Goldberg, *Introduction to Digital Audio Coding and Standards*, (Kluwer Academic Publishers, 2003).
5. Altera Corporation, *Stratix II Device Handbook*, (2006). `http://www.altera.com`
6. Altera Corporation, *Quartus II Development Software Handbook v6.0*, (2006). `http://www.altera.com`
7. ARM Limited, *ARM940T Technical Reference Manual*, (2000). `http://www.arm.com`
8. T.-H. Tsai, Y.-C. Yang, *Low power and cost effective VLSI design for an MP3 audio decoder using an optimized synthesis-subband approach*, in: IEE Proc. Comput. Digit. Tech., vol. **151**, (2004).
9. Tensilica Inc., *Audio for Xtensa HiFi 2 audio engine and Diamone 330 HiFi*, (2007). `http://www.tensilica.com`
10. VLSI Solution Oy, *VS1011e - MPEG AIDIO CODEC*, (2005). Datasheet: `http://www.vlsi.fi`, 2005
11. S. Hong, D. Kim and M. Song, *A low power full accuracy MEPG1 audio layer III (MP3) decoder with on-chip data converters*, in: IEEE Transactions on Consumer Electronics, vol. **46**, (2000).
12. Atmel Corporation, *Single-chip MP3 decoder with full audio interface*, (2006). Datasheet: `http://www.atmel.com`